

# Your Are “Proxy” And I Know It

<https://isthatfreeproxy-safe.com/>

Diego Perino, Claudio Soriente, Matteo Varvello  
Telefónica Research

## ABSTRACT

Web proxies provide anonymity, censorship circumvention, etc. Many of these web proxies are free of charge despite offering valuable services to thousand of users. We suspect that these free web proxies might be performing malicious activities, such as content injection or user tracking, to fund their free service. We thus built a measurement platform that discovers free proxies via web crawling, and tests each proxy to assess its performance and behavior. We collect data daily and make it available at [4]. We here present an analysis of one week worth of data spanning 70,000 free HTTP proxies. Our analysis shows that free proxies are “free for a reason”.

## 1. INTRODUCTION

Web proxies are intermediary boxes enabling an HTTP (sometimes also HTTPS) connection between a client and a server. Web proxies are widely used for security, privacy, performance optimization or policy enforcement, to cite a few use cases. Many web proxies are free of charge and publicly available. Such proxies are mostly used for private web surfing and to access content that would be otherwise blocked (e.g., due to geographical restrictions).

Free web proxies are easy to find. Thousands of free proxies are periodically announced on specialized forums. Even professional VPN service providers like `hidemyass.com` distribute daily updated lists of free web proxies. Free web proxies are also widely used. Recent work [7] has found more than 50k users per week aggregated over 50 proxies and estimates that free web proxies handle roughly 220TB of daily traffic. Yet, the business model of free web proxies is unclear since, apparently, they provide a free, valuable service in exchange of nothing.

Similar to proxies (or middleboxes) set up by ISPs, free web proxy can read, block or alter any traffic between two endpoints that go through the proxy. Previous work [8] has found that traffic manipulation by ISP proxies is carried out for performance/security (e.g., image-size reduction, malware protection, etc.) as well as for malicious goals (e.g., ad injection, fingerprinting, etc.). Nevertheless users cannot avoid ISP proxies. Simply by using an ISP, a user is exposed to any traffic manipulation carried out by the proxies of that ISP. Differently, free web proxies are explicitly configured by users in their browsers, usually to afford anonymous surfing or to access content that is not available at the user’s lo-

cation. By going through a free web proxy, however, users entrust all their internet activity to the proxy and all their traffic gets exposed to eavesdropping and manipulation by a potentially prying service.

To the best of our knowledge, no systematic study of the behavior of free web proxies exists and we are the first to try to shed light on their trustworthiness. We introduce a measurement platform that assesses both performance and behavior of free web proxies announced on the web. We collect data daily and make it available at [4]. We also present an analysis of one week worth of data spanning up to 70,000 free web proxies. Results show that the large majority of the announced proxies are either unreachable or do not proxy traffic. Half of the working proxies appear as non-malicious but exhibit bad performance. Conversely, many of the best performing proxies show malicious activities from malware injection to client fingerprinting or man-in-the-middle attacks. This preliminary analysis suggests that free web proxies are “free for a reason” and potentially match good performance with malicious behavior.

## 2. BACKGROUND AND RELATED WORK

A *web proxy* is a device/application that acts as an intermediary for HTTP(S) requests, such as GET and CONNECT, issued by clients seeking resources on servers. The CONNECT method [3] is an extension to HTTP/1.x made to allow web proxies to tunnel encrypted requests via TLS. By default, the proxy establishes a TCP connection to the specified server, responds with an HTTP 200 (Connection Established) response, and then forwards data between the client and the server, without visibility of the tunneled traffic.

Web proxy can be classified as *transparent*, *anonymous*, and *elite*, depending on the degree of anonymity they provide to their clients. Transparent proxies reveal the IP address of the client to the origin server, e.g., by adding extra headers to the HTTP request. For example, some transparent proxy add the X-FORWARDED-FOR specifying the address of the client. Anonymous proxies block headers that may allow the origin server to detect the identity of the client, but still announce themselves as proxies, e.g., by adding the HTTP\_VIA header. Elite proxies do not send any of the above headers and look just like regular clients to the origin server. Yet, the origin server may check if a proxy is being used by other means such as browser fingerprinting.

**Related Work.** The authors of [6] use `javascript` at the client to detect en-route modifications to a webpage. They report 650 cases of content manipulation over 50k requests. Similar to our findings, the results reported in [6] are a lower bound as 3rd parties may modify only specific content, e.g., pages served from a given domain.

The authors of [7] conducts a measurement study of open proxies to characterize how much these systems are used, what they are used for, and who uses them. They were able to provide some of the results (e.g., search query statistics) because the management interface of many of the proxies included in the study, was reachable without authentication.

The only publicly available tool which shares similar capabilities with our work is `proxycheck`.<sup>1</sup> This tool downloads few distinct objects hosted on a private webserver via a proxy, and considers the proxy untrusted if the retrieved objects differ from the original ones. While an interesting first step, `proxycheck` generates a large number of false positives as it does not distinguish partial downloads or hosts that do not proxy from actual suspect activity. Also, it does not provide any information about proxy performance and it is not scalable ( $\sim 10,000$  proxies can be evaluated per day).

### 3. METHODOLOGY

Monitoring the free web proxy ecosystem is challenging. On the one hand, we need to devise a methodology without really knowing which malicious behavior we are looking for. On the other hand, such methodology needs to be scalable to quickly identify potentially malicious proxies.

Our intuition is that such proxies might pursue malicious activities while offering a free service, e.g., infect clients to create botnets or collect and sell user information for advertisement. We thus opted for instrumenting a regular client and fetch content both with and without proxy interception. We use content served on a website under our control to be able to insert “bait” code. For example, we can add (fake) advertisement via `javascript` and observe whether the proxy tries to replace it with its own advertisement.

To tackle scalability, we split proxy testing into two tasks that can run concurrently. The first task is responsible to identify proxies that are reachable and properly working. This task operates on a large set of proxies (tens of thousands) and thus needs to be quick. The second task extensively tests the subset of proxies indicated as properly working. This task operates on a potentially smaller set (few thousands) and it thus has more time available per proxy.

In the remainder of this section, we detail our monitoring platform as well as how we identify potentially malicious proxies. Finally, we discuss a set of limitations.

#### 3.1 Monitoring Platform

Our monitoring platform has two main components: a web server and a testing client. We use `nginx`<sup>2</sup> as web-server implementation to host *synthetic* content for testing.

The web server is hosted in Ireland by Amazon Web Services and supports both HTTP and HTTPS using standard X.509 certificate.<sup>3</sup> The testing client is a regular Linux machine (Ubuntu version 14.04) located at our lab in Barcelona (Spain), equipped with a 50 Mbps fiber connection. The data collected and analyzed is daily reported at [4].

The synthetic content consists of a simple 1KB page with binary data, and a realistic website. The realistic website is designed to include elements that could trigger content manipulation by a proxy. In particular, we use a simplified clone of the Wikipedia landing page made of `index.html` (83.7KB), two `javascripts` (635B and 22.9KB), two `png` images (1.5KB and 13.5KB) and a `favicon` (4.3KB). We also add a fake advertisement which triggers the download of `adsbygoogle.js` despite not showing any real ads.

Monitoring operations are structured in three phases:

**Phase I:** It discovers, daily, free proxies (`<ip, port>` pairs) listed on the web. We have identified three popular aggregator websites which regularly update free proxy lists. We crawl each aggregator via a python script built around the `beautifulsoup` library.<sup>4</sup> We then aggregate the data collected into a unique “proxy list”. We recognize that many more proxies exist on the Internet and that one could also discover proxies by probing well-known ports (e.g., 3128 or 8080) of arbitrary IP addresses. However, our goal is to test proxies that the average user would find via a simple Google search. Our rationale is that a malicious proxy hunting for victim clients will be well advertised on most of the aggregators websites. Apart from the set of daily proxies fetched from aggregators, we bootstrap our proxy list using large sets of proxy addresses obtained from several specialized forums.

**Phase II:** It quickly tests the full proxy list and categorizes proxies as *unresponsive*, *good*, and *not-a-proxy*. We use `curl`<sup>5</sup> to attempt fetching the simple 1KB page hosted at our server. We instrument `curl` for full statistics and headers collection, as well as with TCP connection timeout (`-connect-timeout`) and global timeout (`-max-time`) to avoid getting stuck on unresponsive proxies. (See Section 4.1.) Next, we compare the received content, if any, with the content received when no proxy was used. An alternative approach is to compare the content received with the one we host at our server. We used the former approach to rule out content manipulation by a transparent proxy (e.g., the one of an ISP) that may be close to our client. We label as *unresponsive* the `<ip, port>` pairs for which either a connection or max duration timeout was triggered. We label as *good* all the `<ip, port>` pairs which returned at least 50% of the requested content. Finally, we label as *not-a-proxy* all the remaining `<ip, port>` pairs whose content returned largely departs from the requested one. (For example, many advertised proxies simply return a login page to the proxy platform.)

**Phase III:** It extensively tests good proxies to estimate their performance and behavior. We use Google Chrome to fetch our realistic website via each good proxy, as well as without a proxy. We choose Chrome, instead of headless browsers

<sup>1</sup>[https://github.com/chrisiaut/proxycheck\\_script](https://github.com/chrisiaut/proxycheck_script),  
<https://proxycheck.haschek.at/>

<sup>2</sup><https://nginx.org/>

<sup>3</sup>We use a free CA available at <https://letsencrypt.org/>

<sup>4</sup><https://launchpad.net/beautifulsoup/>

<sup>5</sup><https://curl.haxx.se/>

like PhantomJS,<sup>6</sup> to be as realistic as possible. Precisely, we use Chrome’s remote debugging protocol to extract an HTTP Archive (HAR). The HAR file includes detailed information about which object was loaded when, and when the page was fully loaded (on-load event). We stop Chrome either one second after the onload event, to allow for potentially pending objects to be downloaded, or after a global timeout. We use Xvfb<sup>7</sup> (the X virtual frame buffer) to avoid the need of a real monitor. We also capture videos of website loading to compare the actual visual output with the expected one. Finally, in the background we attempt fetching the simple 1KB page using curl and HTTPS to investigate potential man-in-the-middle attacks.

### 3.2 Behavioral Analysis

Phase III data is used to improve upon the *not-a-proxy* labeling from phase II. We leverage the videos of the website loads to identify content which is visually different than expected, *i.e.*, 90% or more pixel difference with the page fetched when no proxy was used. Next, we further classify proxies as: *trusted*, *untrusted*, and *too-slow-to-judge*.

**Trusted:** A proxy serving the expected content, *i.e.*, no difference between the set of files retrieved via the proxy and the set of files retrieved when no proxy is involved.

**Untrusted:** A proxy that either alters our content, adds unsolicited content, inserts tracking cookies, or all of the above.

**Too-slow-to-judge:** A proxy that operates at a very slow speed and it is incapable to serve the full requested content (*i.e.*, missing or incomplete files). It follows that the partial content received is not enough to judge its activity in most cases. However, we classify the proxy as **untrusted** if it exhibits such behavior even with just a partial download.

### 3.3 Limitations

We currently only use synthetic content. This is not an intrinsic limitation of our system, as we could easily point the test client towards real content. Real content is interesting since it allows to observe a proxy in its natural environment. However, real content can be complex and dynamic making it hard to evaluate content manipulation. Accordingly, rather than extending our methodology with real content we plan to further increase the amount and diversity of synthetic content we serve by scraping and serving real websites.

Our methodology may also exhibit false negatives since a proxy may choose to only alter certain pages, excluding those hosted at our server, or it may choose to modify content only for certain clients, excluding our test client. It follows that our results are a lower bound on the number of free web proxies that are potentially malicious. To limit the number of false negatives, we plan to distribute our client over multiple vantage points. This approach also allows to investigate the impact of geolocation on proxy behavior which we currently do not explore.

We defer an in-depth analysis of the injected content to future work and remark that analyzing opaque content (e.g., obfuscated javascript) is a research challenge on its own [2].

<sup>6</sup><http://phantomjs.org/>

<sup>7</sup><https://www.x.org/archive/X11R7.7/doc/man/man1/Xvfb.1.xhtml>

## 4. RESULTS

This section presents the analysis over one week worth of data (09/09/2016–09/16/2016) collected by our measurement infrastructure. Fresh results are also updated daily at [4]. We start by calibrating our measurement platform with respect to the timeout selection problem (Section 3.1). Next, we dive into the analysis of the free web proxy ecosystem in terms of availability, performance, and overall behavior. When possible, we compare our results with the ones we obtained by running proxycheck.

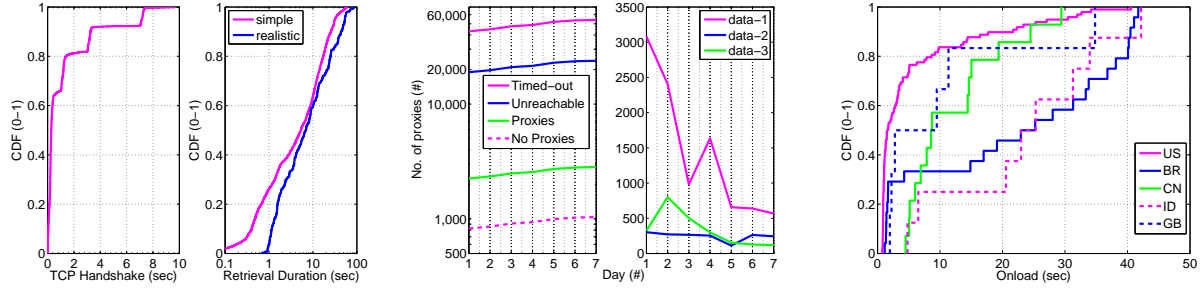
### 4.1 Timeout Selection

We measure TCP handshake duration as well as content retrieval—both for the 1K object using curl and the realistic webpage using Chrome—for the 57,000 proxies available at bootstrap in our proxy list. For this measurement, we allow very large timeouts of respectively 10 seconds (TCP handshake) 60 seconds (1KB download) and 90 seconds (realistic website fetch). Figure 1(a) shows the Cumulative Distribution Function (CDF) of the duration of TCP handshake, 1k object retrieval (labeled “simple”), and page load time of a realistic website (labeled “realistic”). The page load time (PLT) is computed via the classic browser’s “onload” event [1]. The plot refers to the 3,000 proxies out of 57,000 that were reachable (Section 4.2).

Figure 1(a) (left hand-side) shows that 65% of the TCP handshakes complete within one second, *i.e.*, the default TCP retransmission timeout (RTO).<sup>8</sup> Next, TCP retransmits the SYN packet which succeeds for an additional 17% of the proxies (TCP handshake duration lower than 3 seconds). At 3 seconds, a second (doubled) RTO occurs and a new SYN retransmission occurs. In this case, only an handful of proxies further complete the TCP handshake before the next RTO occurs, at 7 seconds. Instead, 10% of the proxies eventually succeed after another SYN retransmission and have an overall TCP handshake lasting between 7 and 10 seconds, the very large TCP handshake timeout we set for this test.

Figure 1(a) (right hand-side) shows the CDF of the time needed to retrieve a simple object and a realistic website for the 3,000 reachable proxies. When no proxy is used, the 1KB object can be fetched in  $\sim 100$ ms and the realistic website in  $\sim 400$  ms. If we focus on the “simple” curve, we notice how 94% of the proxies can complete the 1KB download within 30 second, and the remaining 6% have download time of up to 60 seconds. These 6% of the proxies had TCP handshakes lasting more than 7 seconds (Figure 1(a), left hand-side). Based on these observations, we set the TCP connection timeout (`--connect-timeout` in curl) to 3 seconds, and the maximum duration to 30 seconds (`--max-time` in curl). We increase such timeout to 45 seconds for the realistic content retrieval via Chrome which covers 92% of the reachable proxies.

<sup>8</sup>The test client runs kernel 3.13.0 which implements TCP Cubic with an initial RTO of 1 second [5].



(a) CDF of TCP handshake and content retrievals ; 57,000 proxies. (b) Time evolution of proxies by category. (c) Page load time by proxy's country.

Figure 1: Data Analysis.

## 4.2 Availability

Figure 1(b) (left hand-side) shows the evolution over one week of the number of proxies we tested divided as unresponsive (timed-out and unreachable), good, and not-a-proxy. The majority of the proxies (66%) fail because of a TCP timeout<sup>9</sup> which further strengthen the need of a proper setup as discussed above. Next, 30% of the proxies are “unreachable”, *i.e.*, they either close the TCP connection with a reset message or send ICMP messages declaring the network or host unreachable. Only 3% of the proxies are indeed real proxies, *i.e.*, they serve most of the content requested, while the last 1% are “not-a-proxy” since they serve some completely unrelated content. Note that unreachable proxies could be labeled as not-a-proxy as well. However, we divided them because a subtle difference exists. While unreachable proxies might end up by mistake in public proxy lists, the behavior of “not-a-proxy” could be considered suspicious: in fact they could leverage public proxy lists to attract traffic and boost their popularity, click rate, etc.

The amount of proxies we test constantly increase due to our platform daily pulling new proxies from public sources.

## 4.3 Performance

Figure 1(c) shows the CDF of the PLT as a function of the country where the proxy is located. Proxy location is an important attribute as many proxies are used to access geo-restricted content. For the sake of visibility, the figure only reports data from the five countries where most proxies are hosted: United States (42%), Brazil (11%), China (8%), Indonesia (5%), and United Kingdom (4%). The remainder of the proxies are located in 28 countries. Location information is derived via ip geolocation using Maxmind.<sup>10</sup>

Overall, Figure 1(c) shows that finding good performing proxies in the US is feasible: 50% of the proxies provide a PLT lower than 1.5 second, which is only 1 second slower than accessing the page directly. Similarly, 20% of proxies in United Kingdom and Brazil provide PLT smaller than 3 seconds, *i.e.*, a tolerable user experience. For the remainder of the countries (and proxies), we measure extremely high PLT up to the 45 seconds global timeout. Clearly, these PLT values imply practically unusable proxies.

<sup>9</sup>Only 1% fail due to the global timeout.

<sup>10</sup><https://www.maxmind.com/>

## 4.4 Behavior

Table 1 summarizes the behavioral analysis of the working proxies. We distinguish between “too-slow-to-judge” (slow) proxies and fast ones, or proxies that manage to complete the requested content retrieval. Results are averaged over the 7 days period. As the table shows, 2% of the proxies either inject suspicious code in the requested content (65% of activities), or add extra content whose download is triggered by index.html modification (35% of the activities). These malicious proxies tend also to be among the fastest ones, *i.e.*, they provide a PLT lower than 3 seconds.

The table also shows that the presence of our bait advertisement (WithAds column) causes an increase of the “too-slow-to-judge” proxies; 43% less proxies complete their job when advertisement must be served. On the one hand, this is due to additional content to be fetched by already poorly performing proxies. On the other hand, it is possible that the javascript associated to our bait advertisement is directly delayed by Google—though not verifiable without actual access to the proxy.

For comparison, we run proxycheck over three days and it was able to check about 23K proxy only. Results report about 1,600 active proxies, the same amount we found over 70K proxy evaluated, 80% of them being untrusted. This clearly highlights the significant false positive rate of proxycheck in terms of both active and untrusted proxies, as well as its scalability issues.

To further understand the previous findings and provide insights on proxy behavior, we perform manual inspection of the content retrieved and we manually test a subset of the (more) suspicious proxies. We plan to automate such analysis as part of our future work. This analysis shows that html files are more likely to be modified compared to javascripts and images. Unsolicited content injected by proxies mostly consist of javascript files, though we also spotted php and image injections. No statistically significant cookie injection was observed. In the following, we report examples of malicious activity present in our data.

*Client fingerprinting:* Several proxies inject malicious code in existing pages or unsolicited files which retrieves client information (e.g., OS, browser, computer name, Windows serial number) and upload it to a remote host.<sup>11</sup>

<sup>11</sup><https://goo.gl/R65GUd>

	WithAds		NoAds		Activity	
	Fast	Slow	Fast	Slow	Modif.	Add.
Untrusted	0%	2%	1%	1%	65%	35%
Trusted	37%	61%	80%	18%	-	-

**Table 1: Summary of behavioral analysis.**

*Man-in-the-middle:* One proxy was attempting a man-in-the-middle attack by providing a self-signed certificate. The specific proxy was running GoProxy.<sup>12</sup> Note that this attack has low chances to succeed since it requires the victim machine to accept self-signed certificates, which is not a common practice.

*Malware installation:* Several proxies inject malicious code that retrieves malware from remote machines and installs it on the client host. One specific example we identified is a malware which modifies some local Windows registers.<sup>13</sup>

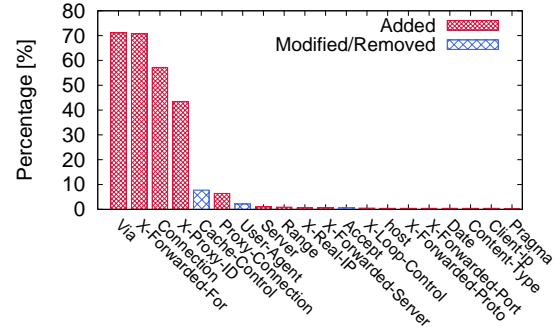
## 4.5 Header Analysis

We now analyze the HTTP request/response headers at client/server sides with the twofold objective of understanding the level of anonymity provided by proxies, and header manipulations they might perform.

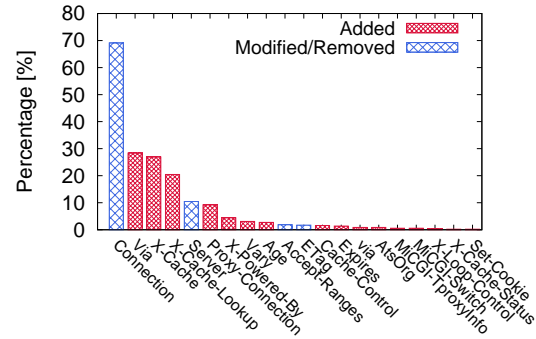
Figure 2(a) shows the top 20 request header modifications and injections performed by proxies. The most frequent injections are the `Via` and `X-Forwarded-For` headers. The former is used by proxies to announce themselves, while the latter is used to specify the original client IP to the server when the proxy acts transparently. Only 30% of proxy are anonymous or preserve client anonymity. Another frequently added header is `Connect`. As we do not specify any `Connect` header at the client, about 60% of proxies set it to `close` or `keep-alive` values. This is not surprising as those headers should be used for each connection and not communicated by proxy over further connections. `Cache-Control` and `User-Agent` are the most frequent request headers modifications. About 7% of proxies modify headers to accept cached content with a given `max-age` value despite we explicitly specify not to serve cached content. We also observe that only 2% of proxies modify the user-agent by either removing it or specifying their own agents. This is “good” since it implies that the server can provide the most appropriate content to its client. However, it reduces anonymity since it exposes the original client user-agent.

Figure 2(b) shows the top 20 response header modifications and injections performed by proxies. As for the request headers, the `Via` header is the most frequent injection; this is used by proxies to announce themselves to clients. About 30% of proxies add the `X-Cache` and the `X-Cache-Lookup` headers. These headers are used to specify if the requested content was served from the proxy cache, and if a cacheable response is available in the proxy cache. The most frequently modified header is the `Connection` header, that is either removed (50% of cases) or set to `close`. As highlighted above, this is a common behavior as those headers are connection specific and do not need to be propagated to the server. Finally, only 10% of the proxies modify the `Server`

header to reflect the software they use, rather than the original server software.



(a) Requests.



(b) Responses.

**Figure 2: Header analysis.**

## 5. CONCLUSION AND FUTURE WORK

This paper has presented a methodology and a live platform to analyze performance and behavior of free web proxies. A preliminary analysis over a week-worth of data shows that only few thousand proxies (out of 70,000 reported on the web) are real proxies. The remainder are mostly unreachable machines or not proxy at all. Real proxies tend to be very slow but non-malicious, with only a small percentage showing suspicious behaviors like content injection (ads and malware) and page alteration. These suspicious proxies offer the best downloading speed, potentially a strategy to attract their victims.

Our avenue for future work is threefold. First, we will extend our methodology to cover additional free proxies and other anonymizing infrastructures like Hola (<http://hola.org/>), TOR (<https://www.torproject.org/>) or UProxy (<https://www.uproxy.org/>). Second, we will extend the footprint of our infrastructure to cover distinct network locations as well as to scale with a constantly growing proxy set. Finally, more work needs to be done to understand the business model of free web proxies and the entities behind them.

<sup>12</sup><https://github.com/elazarl/goproxy>

<sup>13</sup><https://goo.gl/hPPRFY>

## 6. REFERENCES

- [1] Globaleventhandlers.onload.  
<https://developer.mozilla.org/en/docs/Web/API/GlobalEventHandlers.onload>
- [2] COVA, M., KRÜGEL, C., AND VIGNA, G. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th International Conference on World Wide Web, WWW* (2010), pp. 281–290.
- [3] KHARE, R., AND LAWRENCE, S. RFC2817: Upgrading to TLS Within HTTP/1.1, 2000.  
<https://tools.ietf.org/rfc/rfc2817.txt>.
- [4] MONITORING, P. A platform for monitoring proxy safety and performance.  
<https://isthatfreeproxySAFE.com/>.
- [5] PAXSON, V., ALLMAN, M., CHU, J., AND SARGENT, M. RFC 6298: Computing TCP’s Retransmission Timer, 2011.  
<https://www.ietf.org/rfc/rfc6298.txt>.
- [6] REIS, C., GRIBBLE, S. D., KOHNO, T., AND WEAVER, N. C. Detecting in-flight page changes with web tripwires. In *Proc. NSDI* (San Francisco, CA, USA, Apr. 2008).
- [7] SCOTT, W., BHORASKAR, R., , AND KRISHNAMURTHY, A. Understanding open proxies in the wild. In *Chaos Communication Camp* (2015).
- [8] WEAVER, N., KREIBICH, C., DAM, M., AND PAXSON, V. Here be web proxies. In *Passive and Active Measurement - 15th International Conference, PAM* (2014), pp. 183–192.